Assignment 7

In this assignment we will work on two languages: "L-values" and "S-expressions".

L-values are used to express the left-hand-side of an assignment in programming languages that support that (e.g. `a[i][j] = 2` in C). Surprisingly, the "L" does not stand for "left", but instead for "location", meaning that these "values" represent locations in memory, whose contents are about to be replaced.

S-expressions are used as the main syntactic structure in languages like Lisp, Scheme and Racket. They are probably the simplest syntax for a programming language, in terms of parsing and number of rules.

The grammar of L-values has the terminals `v`, `i`, `[` and `]`. Its main non-terminal will be denoted by `L`, so one of the grammar rules would be `S–>L`. You will need one more nonterminal. An L-value is one of the following:

- A variable name, in general indicated by the terminal symbol `v`.
- An expression `l1[l2]`, where `l1` is an arbitrary L-value and `l2` is an "index" which can be either an arbitrary L-value or an integer. Integers are denoted generically by the terminal symbol `i`. You should use a new nonterminal, denoted by `I`, for "index".

Here is an example expression that your grammar should be able to pick up: `v[v[i][v]]`.

Full-fledged L-values are more complex, but this will do for our purposes.

The grammar of S-expressions has terminals `a`, `(` and `)`. The main terminal for an S-expression will be denoted by `E`, so one of the rules will be `S–>E`, and you will need one more non-terminal. An S-expression is one of the following:

- An "atomic value", indicated by the terminal `a`,
- An expression `(...)` where the dots contain one or more S-expressions. You should use a new terminal `T` to denote this potential list of S-expressions. The elements on that list are meant to be evaluated in a left-to-right way (left-associative so to speak). You should ensure that your grammar rules reflect that.

Here are the questions regarding these two languages.

1. For the L-value language:
   a. Produce the CFG
   b. Compute the first sets for all nonterminals. Explain your work.
   c. Compute the follow sets for all nonterminals. Explain your work.
   d. Construct the DFA of item sets for the LR-parser that corresponds to this grammar.
   e. Show how the L-value `v[v[i][v]]` will be processed by this parser. The format for that would be in 3 columns, one for the input changes, one for stack contents with DFA numbers included, and a third for what happens at each step (shift/go to a state, reduce a grammar rule).

2. For the S-expression language:

   a. Produce the CFG
   b. Compute the first sets for all nonterminals. Explain your work.
   c. Compute the follow sets for all nonterminals. Explain your work.
   d. Construct the DFA of item sets for the LR-parser that corresponds to this grammar.
   e. Show how the S-expression (a(a(aa)a)) will be processed by this parser. The format for that would be in 3 columns, one for the input changes, one for stack contents with DFA numbers included, and a third for what happens at each step (shift/go to a state, reduce a grammar rule).